LAB 9

Objective: Deploy an Ingress controller, create an Ingress rule to reach a clusterIP type service.

The manifest for an Ingress controller that works on **minikube** is in your lab folder (retrieve from *Course Resources*) and is called **backend.yaml**. Just create the controller with **kubectl**:

```
$ kubectl create -f backend.yaml
```

With the Ingress controller now running, you will repeat part of the lab from Chapter 8 and create an Ingress rule as well.

To make this straightforward, check the manifest **nginx.yaml**. You will see the manifest for a Pod, a service that matches the pod labels, and an Ingress rule. Create it.

```
$ kubectl create -f nginx.yaml
```

You will now see a running pod, a service, and an Ingress rule.

```
$ kubectl get ingress
NAME      HOSTS                          ADDRESS          PORTS     AGE
nginx     nginx.192.168.99.100.nip.io    192.168.99.100   80        3m
```

The manifest describing the Ingress is shown below:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: nginx
spec:
```



```
rules:
- host: nginx.192.168.99.100.nip.io
  http:
    paths:
    - backend:
        serviceName: nginx
        servicePort: 80
```

This Ingress rule is used by the Ingress controller (started by the **backend.yaml** manifest) to re-configure the nginx proxy running on the head node (in our case, **minikube**). The rule will proxy requests for host **nginx.192.168.99.100.nip.io** to the internal service called **nginx**.

We use the nip.io service. It is a wildcard DNS service that is very handy for testing. It will resolve **nginx.192.168.99.100.nip.io** to **192.168.99.100** the IP of minikube. Note that you may need to edit the Ingress rule manifest if the IP of your minikube is different.

Once the rule is implemented by the controller (could take O(10) s), open your browser at **nginx.192.168.99.100.nip.io** and enjoy **nginx**.

Try to reproduce this with a **ghost** application.

